- **Bayesian linear regression and Ridge regression**
  In the Bayesian linear regression settings, when $\mathbf{w}_0 = \mathbf{0}$ and $S_0 = \tau^2\mathbb{I}$, we have:

  $$\ln p(\mathbf{w}|\mathbf{t}) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(t_n - \mathbf{w}^t\phi(\mathbf{x}_n))^2 - \frac{1}{2\tau^2}||\mathbf{w}||_2^2$$

  In this case, MAP, maximum a-posteriori, ($\mathbf{w}_N$) is equivalent to the solution of ridge regression ($\hat{\mathbf{w}}_{ridge}$) with $\lambda = \frac{\sigma^2}{\lambda^2}$. Bayesian regression also allows to account for regularization (which can be obtained as a specific case).

- **k-Nearest Neighbor**
  To classify a point $\mathbf{x}$: select the k most similar points to $\mathbf{x}$ in the training set and assign to $\mathbf{x}$ the most frequent class among those. No actual model is computed: we use the training records to predict an unknown class label. The elements are: the training dataset, the similarity function and the value of k.

  1. Compute distance to other training records
  2. Identify the k nearest neighbors
  3. Use labels of nearest neighbors to determine the label of the point (majority voting/others)

  How many neighbors? It's an hyperparameter (cross-validation changing k at every iteration or nested cross-validation). If k is too small the classification might be too sensitive. If k is too large the neighborhood may include quite dissimilar points.

consistent learners: $N \geq \frac{1}{\varepsilon}\left(8\, VC(H) \log_2\left(\frac{13}{\varepsilon}\right) + 4 \log_2\left(\frac{2}{\delta}\right)\right)$

general learners: with prob $\geq (1-\delta)$ every $h \in \mathcal{H}$ satisfies
$$error_{true}(h) \leq error_D(h) + \sqrt{\frac{VC(H)\left(\ln\left(\frac{2N}{VC(H)}\right)+1\right) + \ln 4/\delta}{N}}$$

- MDP: agent-enviroment interface (well defined states, actions, rewards)
- MDP: Bellman equations (expectation, optimality) to evaluate and find the optimal value functions/ policy
- Computationally unfeasible
- Dynamic programming: approximate the optimal value function/ policy iteratively (policy iteration, value iteration)
- Real application: unknown states, actions, rewards
- Monte Carlo methods: learn value functions/ policy through experience (data: complete episodes of states, actions, rewards)
  - policy evaluation for the evaluation of V_π
  - policy improvement? V_π useless, need for Q_π
  - Q_π evaluation: need for exploration
  - explorating starts, ϵ-soft

	Bellman expectation equation	Bellman optimality equation
Dynamic Programming	• policy iteration	• value iteration
Empirical version	• Monte Carlo control • SARSA	• Q-learning

actual class :

		1	0
predicted :	1	tp	fp
	0	fn	tn

- accuracy: $Acc = \frac{tp+tn}{N}$
- precision: $Pre = \frac{tp}{tp+fp}$
- recall : $Rec = \frac{tp}{tp+fn}$
- F_1 score: $F_1 = \frac{2 \cdot Prec \cdot Rec}{Prec + Rec}$

SVM

Separable

minimize $\frac{1}{2}\|\underline{w}\|_2^2$
s.t. $t_n(\underline{w}^T\phi(\underline{x}_n)+b) \geq 1$

maximize $\sum_n \alpha_n - \frac{1}{2}\sum_n\sum_m \alpha_n\alpha_m t_n t_m k(\underline{x}_n, \underline{x}_m)$
s.t. $\alpha_n \geq 0$
$\sum \alpha_n t_n = 0$

$y(\underline{x}) = \sum_n \alpha_n t_n k(\underline{x}, \underline{x}_n) + b$

$b = \frac{1}{|S|} \sum_{\underline{x}_n \in S} \left(t_n - \sum_{\underline{x}_m \in S} \alpha_m t_m k(\underline{x}_n, \underline{x}_m)\right)$

Non-separable

minimize $\frac{1}{2}\|\underline{w}\|_2^2 + C\sum_n \xi_n$ error (penalties to the violation)
s.t. $t_n(\underline{w}^T\phi(\underline{x}_n)+b) \geq 1 - \xi_n \quad \forall n$
$\xi_n \geq 0$

maximize $-\frac{1}{2}\sum_n\sum_m \alpha_n\alpha_m t_n t_m k(\underline{x}_n, \underline{x}_m)$
s.t. $0 \leq \alpha_n \leq \frac{1}{N}$
$\sum \alpha_n t_n = 0$
$\sum \alpha_n \geq \nu$

parameter that controls both the margin errors and the # support vectors :

$0 \leq$ % margin errors $\leq \nu \leq$ % # support vectors ≤ 1